

## Governance

Date published: 2019-12-17

Date modified: 2021-03-25



# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Flink metadata collection using Atlas.....</b>	<b>4</b>
Atlas entities in Flink metadata collection.....	4
Creating Atlas entity type definitions for Flink.....	6
Verifying metadata collection.....	8

## Flink metadata collection using Atlas

In Cloudera Streaming Analytics, you can use Flink with Apache Atlas to track the input and output data of your Flink jobs.

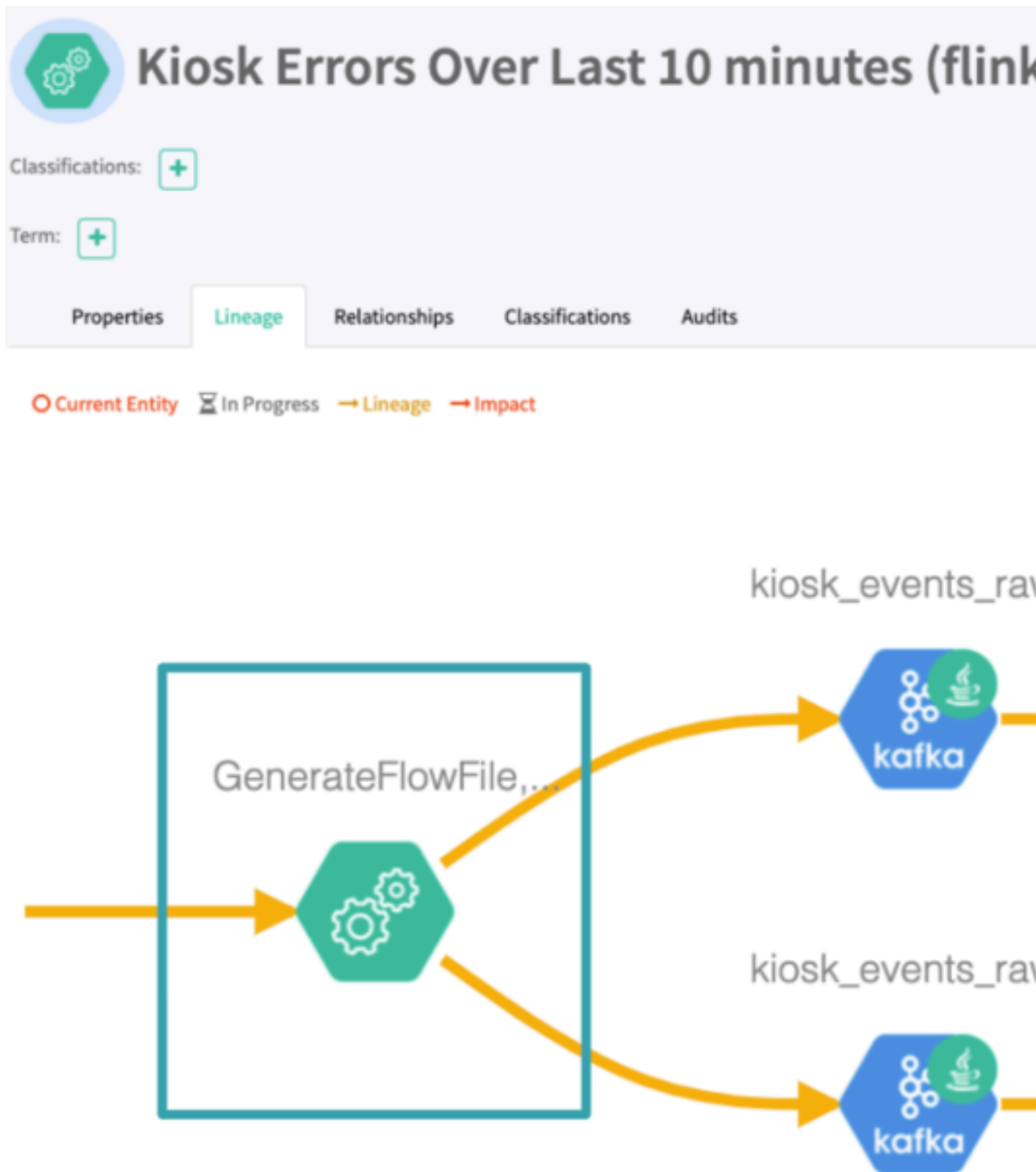
Atlas is a lineage and metadata management solution that is supported across the Cloudera Data Platform. This means that you can find, organize and manage different assets of data about your Flink applications and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

For more information about Atlas, see the [Cloudera Runtime documentation](#).

### Atlas entities in Flink metadata collection

In Atlas, the core concept of representing Flink applications, Kafka topics, HBase tables, and so on, is called an entity. You need to understand the relation and definition for entities in a Flink setup to enhance the metadata collection.

When submitting updates to Atlas, a Flink application describes itself and the entities it uses as sources and sinks. Atlas creates and updates the corresponding entities, and creates lineage from the collected and already available entities. Internally, the communication between the Flink client and the Atlas server is implemented using a Kafka topic. This solution is referred to as Flink hook by the Atlas community.



## Creating Atlas entity type definitions for Flink

Before submitting Flink jobs to collect their metadata, you need to create Atlas entity type definitions for Flink. In the command line, you need to connect to the atlas server and add the predefined type definitions. You also need to enable Atlas for Flink in Cloudera Manager.

### About this task

Atlas does not include the metadata source of Flink by default. The administrator must manually upload the entity type definitions to the cluster to be able to start the Flink metadata collection.



**Note:** The default ports for the Atlas admin server are 31433 and 31000 when TLS is enabled or disabled respectively.

### Procedure

1. Upload the designed entity type definitions to the cluster using the Atlas REST API.

```
curl -k -u <atlas_admin>:<atlas_admin_pwd> --location --request POST 'https://<atlas_server_host>:<atlas_server_port>/api/atlas/v2/types/typedefs' \
--header 'Content-Type: application/json' \
--data-raw '{
  "enumDefs": [],
  "structDefs": [],
  "classificationDefs": [],
  "entityDefs": [
    {
      "name": "flink_application",
      "superTypes": [
        "Process"
      ],
      "serviceType": "flink",
      "typeVersion": "1.0",
      "attributeDefs": [
        {
          "name": "id",
          "typeName": "string",
          "cardinality": "SINGLE",
          "isIndexable": true,
          "isOptional": false,
          "isUnique": true
        },
        {
          "name": "startTime",
          "typeName": "date",
          "cardinality": "SINGLE",
          "isIndexable": false,
          "isOptional": true,
          "isUnique": false
        },
        {
          "name": "endTime",
          "typeName": "date",
          "cardinality": "SINGLE",
          "isIndexable": false,
          "isOptional": true,
          "isUnique": false
        },
        {
          "name": "conf",
```

```
        "typeName": "map<string,string>",
        "cardinality": "SINGLE",
        "isIndexable": false,
        "isOptional": true,
        "isUnique": false
      },
      {
        "name": "inputs",
        "typeName": "array<string>",
        "cardinality": "LIST",
        "isIndexable": false,
        "isOptional": false,
        "isUnique": false
      },
      {
        "name": "outputs",
        "typeName": "array<string>",
        "cardinality": "LIST",
        "isIndexable": false,
        "isOptional": false,
        "isUnique": false
      }
    ]
  },
  "relationshipDefs": []
}
```

2. Log in to Cloudera Manager.
3. Go to **Flink>Configuration**.
4. Search for 'enable atlas' in the search bar.

## 5. Enable Atlas Metadata Collection.

The screenshot shows the Flink UI interface for job **FLINK-1**. The **Configuration** tab is active. A search bar at the top contains the text "enable atlas". Below the search bar, the **Filters** section is expanded, showing a table of configurations under the **SCOPE** filter:

Scope	Count
FLINK-1 (Service-Wide)	1
Flink Dashboard	0
Gateway	0

To the right of the filters, a card titled **Enable Atlas** displays the configuration `atlas.collection.`

### Results

The Flink client notifies Atlas about the metadata of the job on successful submission.

## Verifying metadata collection

After enabling Atlas metadata collection, newly submitted Flink jobs on the cluster are also submitting their metadata to Atlas. You can verify the metadata collection with messages in the command line by requesting information regarding the Atlas hook.

To verify the metadata collection, you can run the "Streaming WordCount" example from Running a Flink Job.

In the log, the following new lines appear:

```
...
20/05/13 06:28:12 INFO hook.FlinkAtlasHook: Collecting metadata for a new Flink Application: Streaming WordCount
```



```
...  
20/05/13 06:30:35 INFO hook.AtlasHook: <== Shutdown of Atlas Hook
```

Flink communicates with Atlas through a Kafka topic, by default the one named ATLAS\_HOOK.

### **Related Information**

[Running a Flink job](#)